**(example title - how should we name it?)**

**Designing of a low-cost Arduino-based Electroencephalography measuring device**

Yago Dórea, Felipe Neves, Leandro Alexandrino and Shin'Ichiro Kanoh.

**Brain-Computer Interface (BCI) is an interface system between a computer and the human brain. It is any form of direct communication from the brain to an output device. Bypassing ordinary body functions such as muscle response or speaking capabilities, it introduces a new way to interact with computers and with ourselves. There are many types of BCI sensors, the simpler ones being based on non-invasive techniques. This technology is rapidly developing in IT recently, but it is still too expensive to buy reliable sensors for research or medical purposes. Non-invasive BCI relies mainly on the electroencephalography (EEG) sensor. The main goal of this project is to develop a reliable EEG sensor that is also relatively cheap. The purpose is to allow more people to be able to conduct research and tinker with BCI.**

**Introduction**

BCI is a technology that allows brain signals to be read and interpreted by a computer. It is a growing field in computer science, and is extremely flexible in it’s possible applications.

Within the field of Human-Computer Interaction (HCI), BCI is an important asset when typical interfaces are not viable. It doesn’t rely on muscular interactions of the user, so it is a new and unique communication channel, allowing new forms of interactions with computers.

By use of this technology, the brain signals of people can be electrically measured and quantified. This information can be then processed, and used to command a computer to perform tasks. This communication bridge is the essence of BCI.

Recently, this field of research has grown a lot, due to advances in signal measuring. Even then, there is a financial barrier to the use and research of BCI, due to the high costs of the adequate measuring equipment. This is also true for applications in the medicine field, for rehabilitation and communication of people with extremely limited mobility.

For instance, there is a health condition named Locked-in Syndrome (LIS), in which the patient is unable to move any part of his body, or communicate in any way. The only way for people with LIS to communicate with other people is by use of BCI. But the medical application of this is still very experimental, and in most cases, extremely expensive.

Electroencephalography (EEG) is a popular non-invasive technique that records brain activity through electrodes in the scalp. The brain processes information through cells called neurons, which communicate electrically with each other through connections known as synapses. Some BCI techniques can measure individual neuron signals by penetrating the skull and placing small electrodes in them. EEG signal, on the other hand, reflects the summation of postsynaptic potentials from many thousands of neurons that are oriented radially to the scalp [1].

The skull, tissues and hair that stand between the source of the electric signals and the measuring electrodes create a high impedance that diminished the spatial resolution of the signal. To treat this problem, some workarounds are introduced, such as a low pass filter, powerful amplifiers and many signal processing techniques. Even then, the signal can be easily contaminated by artifacts, due to its weak amplitude. Eye movements, talking and noises will corrupt the signal generating "muscle artifacts".

All this contributes for EEG measuring devices to be extremely expensive and restricted to the public that can afford such material costs. In Japan, a high-end EEG measurement system is about 12.000.000 JPY (for 80 channels) or around 8.000.000 - 10.000.000 JPY (for 64 channels). That is roughly 115.000 USD and 77.000 - 96.000 USD, respectively.

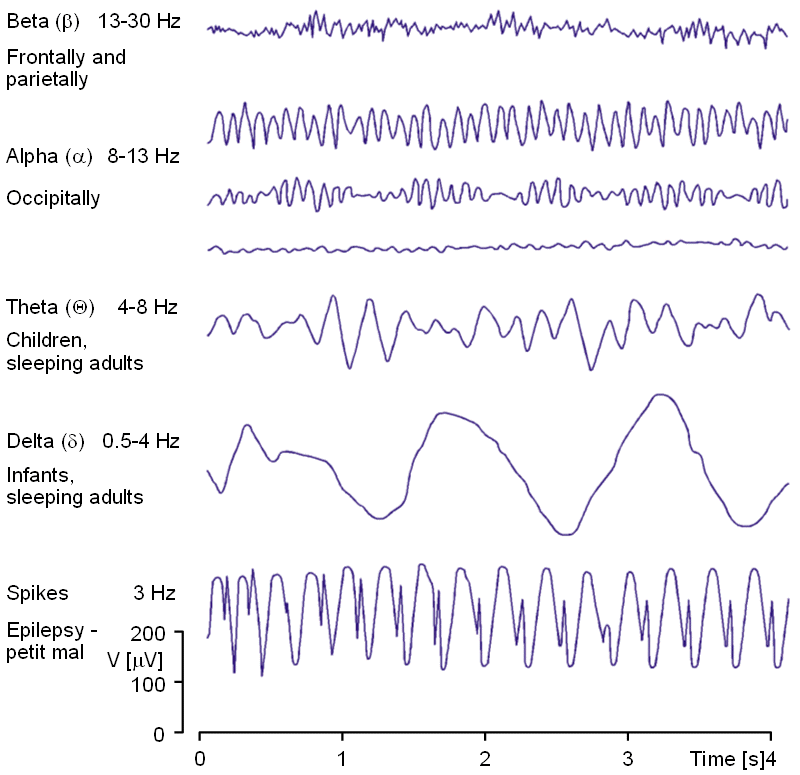
We intend to produce a low-price EEG system using Arduino, low-price electronic amplifiers, and MATLAB processing. We demonstrate here the results of our work, comparing it with a high-end laboratory device.

**Materials and Methods**

*Principles and standards*

The average frequencies measured by EEG can vary from 0.5-4Hz (Delta waves) to around 13-30Hz (Beta waves) (Figure 1).

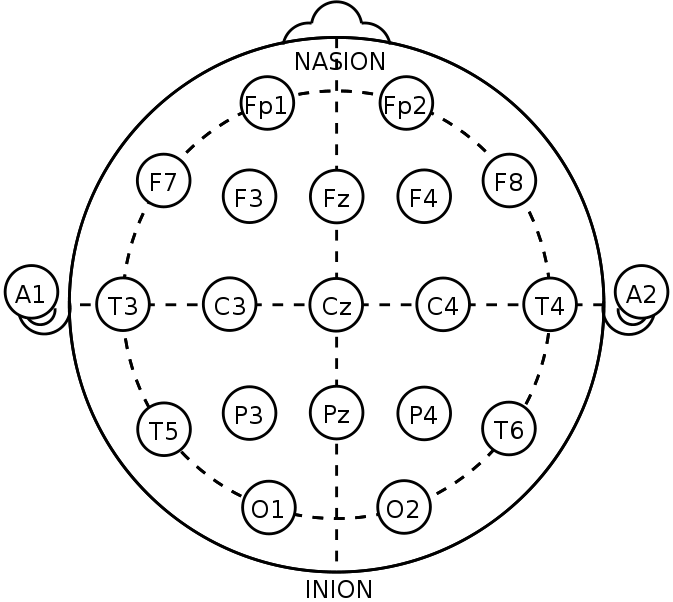
Also, the EEG signal amplitude at the scalp is about 100µV, which means that, to gather data that can be read by an ordinary micro-controller sensor, we have to apply around 10000-50000x amplification (80-94dB).

 EEG measuring has an international standard for electrode positioning over the scalp. It is named the 10-20 system, and it was developed to ensure standardized reproducibility of experiments in scientific community. The locations of the scalp are divided in: C = central, P = parietal, T = temporal, F = frontal, Fp = frontal polar, O = occipital, and A = mastoids (Figure 2).

**Figure 2**: *10-20 system for electrode placing.*

**Figure 1**: *Examples of EEG signals in determined frequency ranges.* (Adapted from http://www.bem.fi/book/13/13.htm).

The mastoids are to be used as reference electrodes. The electrodes are fixed at these points with a conducting paste, and the locations are thoroughly cleaned, to remove dirt, dead tissue and sweat. All that to ensure minimization of signal impedance.



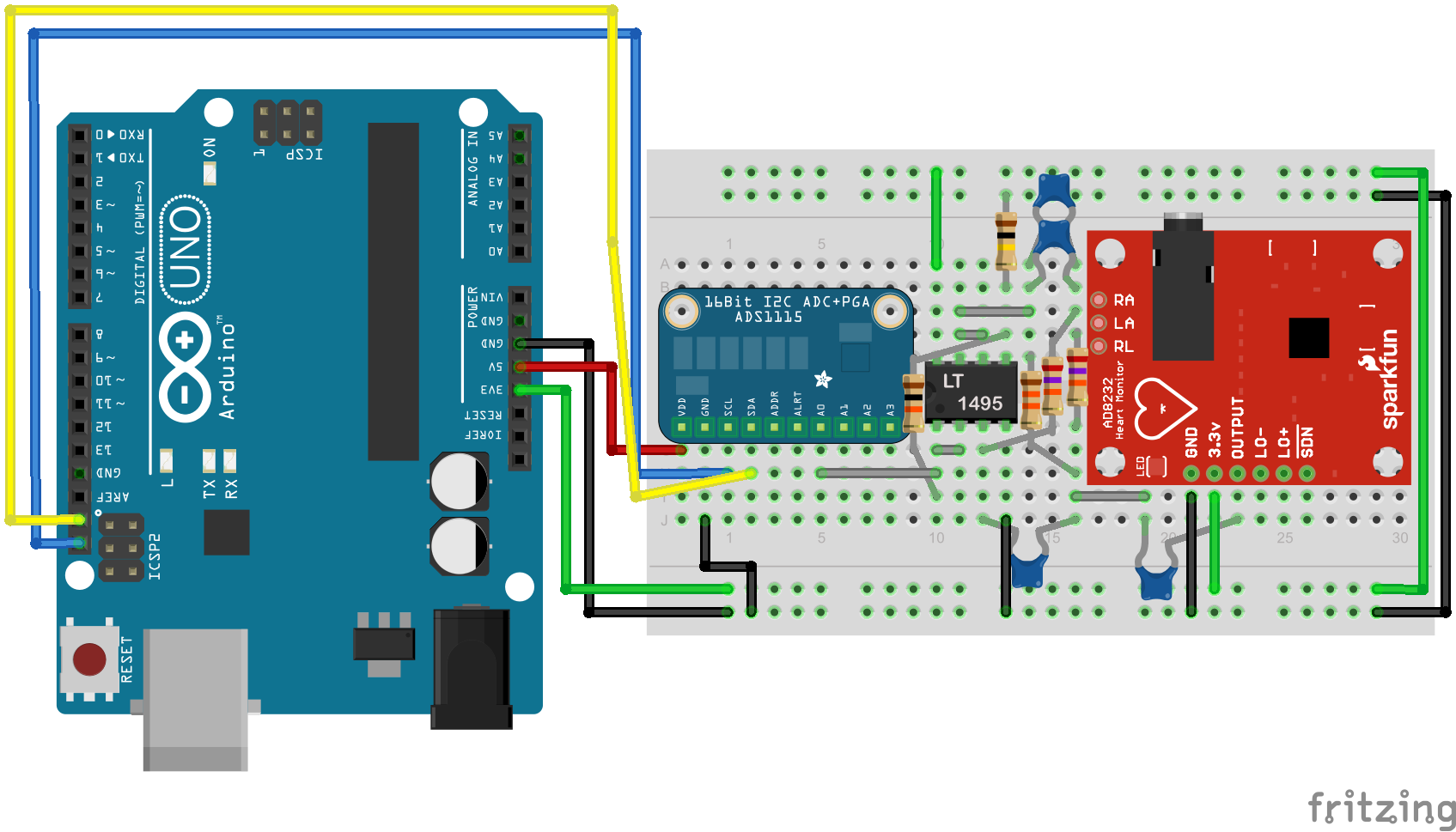
*Materials*

After amplification of the signal, it must be transformed in order to be able to processed by a computer. Thus, a AD (Analog to Digital) converter is needed. In this project, an Adafruid ADS1115 16-bit ADC 4-channel chip was used (for reference: https://www.adafruit.com/products/1085). This chip also includes a programmable 16x amplifier, which will also serve help achieving the project's goal. The conversion can be made in two ways: single-ended input or differential input. When using differential input, only two channels are available at a resolution of 16 bits. On the other hand, by using single-ended inputs, we can get 4 channels at 15-bit resolution. As the ground reference is the same for all four channels, one bit is reserved for zero, so, effectively, only 15 bits are real data.

The sensor used in this project is a SparkFun Single Lead Heart Monitor AD8232 (website for reference - https://www.sparkfun.com/products/12650), which is a sensor for Electrocardiogram (ECG), but can serve the same purpose as a EEG sensor. Three electrodes are wired in it. One for the signal, in the Cz section (see 10-20 diagram), one for reference in one mastoid (A1), and another for ground, in the other mastoid (A2).

Thus, only one channel is being measured for our tests. Subsequent channels can be added using only one electrode per additional section of the scalp, because the ground and reference electrode can be the same. For each additional channel, however, another sensor must be used, as well as another ADC converter for every 4 channels.

**Figure 3**: *Designed circuit for EEG signal sampling*.



The Arduino UNO is a very popular micro-controller used in many types of applications due to its versatility and friendliness: it is a board designed for starters in electronics. It has several digital input/output pins, as well as PWM outputs and analog inputs. It can process data at around 16MHz and communicate via serial with any other device. The data rate in serial communication ranges from 300 baud to 115200 baud (bits per second), while the ADS1115 analog conversion rate is, at maximum, 860 samples per second. As each sample amounts to 16 bits of data, the maximum throughput of this communication will be less than 14000 bits per second, which is supported.

We included a low-power LT1495 amplifier to boost the signal before it is encoded, arranging the circuit to supply around 16.000x amplification of the original signal (Figure 3). The diagram in Figure 4 shows the basic workings of the final device. The signal ins input by the electrodes through the AD8232 sensor, which is then amplified and input to the ADS1115 AD converter. The Arduino captures the data bit by bit at approximately 600 samples per second and feeds the data to the MATLAB through serial connection with a computer (see Figure 4).

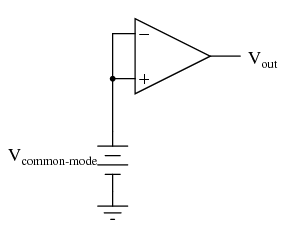
*Matlab and Arduino algothms*

The Arduino uses a specific language named Processing, which is similar to C, but simplified and with specific functions for the microprocessor. The code snippet in this project is simply the acquiring of the ADC signal within a specified time period, so that the reading frequency remains constant. The readings are sent one by one through the serial connection. The MATLAB code captures these samples and arranges them into a vector in the memory. It applies then a low-pass filter to clean the signal and afterwards, a notch filter at 50Hz to filter the 50Hz noise from the power line (as known as "electric hum"). Afterwards, it applies an FFT analysis to identify patterns and perform signal processing.

**Figure 4**: *Diagram showing simplified system.*

*Signal acquiring procedure*

For capturing the signal, the electrodes are placed at the Cz, A1 and A2 areas, for signal, reference and ground, respectively (See Figure 2). The subject is instructed to stay with closed eyes for 10 seconds, and then open them for another 10 seconds.

 Rough body movements, eye movements, and noise in the room may cause artifacts in the signal, and even undesired stimulus-evoked potentials, so it should be kept at minimum.

The capturing of the signal is triggered by MATLAB through the keyboard, and the response can be interpreted on the go, by selecting a small time interval in the sample and analyzing it in the frequency domain, for detection of patterns, type of brain activity, etc.

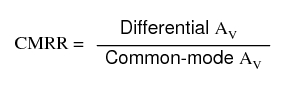
**Results**

We've made a series of tests for analyzing performance and comparing it to high-end laboratory equipment.

**Figure 5**: *Circuit for Acquiring common-mode gain. Vout should remain the same regardless of Vcm. Adapted from: https://www.ibiblio.org/kuphaldt/electricCircuits/Semi/SEMI\_8.html*

*Amplifier evaluation*

eegdevice.png By setting up a specific circuit (See Figure 5), we can measure the amplifier's differential gain (Ad) and it's common mode gain (Ac). The ratio between them is called Common Mode Rejection Ratio (CMRR), and is used to measure the device's capability of rejecting common-mode signals. By definition:



After setting up and measuring the circuit's output, we achieved the gain profile seen in Figure 6.

insert data for closed loop amp tests and such

(CMMR, etc.) (differential/common mode)

*Feature detection*

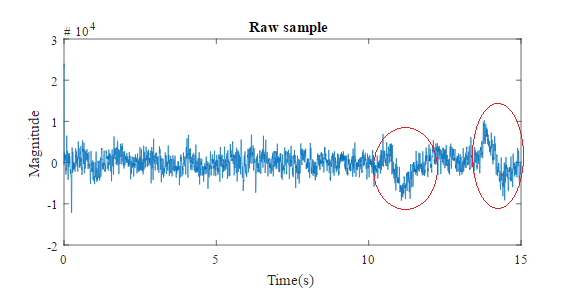
By measuring only one channel, we can still detect specific voltage peaks regarding the blinking of one or both eyes, and muscular stimuli such as raising an arm. Figure 7 shows the impact of the eye blink on the electroencephalogram.

Also, it is possible to detect whether the subject is with closed or opened eyes. By analyzing the sampled signal in the frequency domain, applying a Fourier Transform, we can detect different frequency peaks for "eyes closed" and "eyes opened" (See Figure 8). This is possible because the brain emits different types of brain-waves for different states of brain activity. For instance, it produces higher frequencies when the eyes are opened and there is high activity.

(get Figure 6!)

- with/without electronic filter?

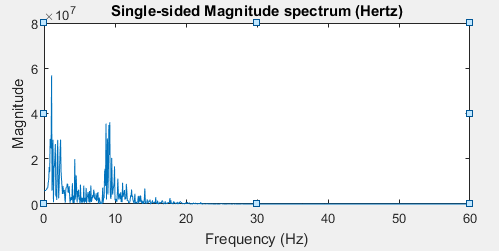
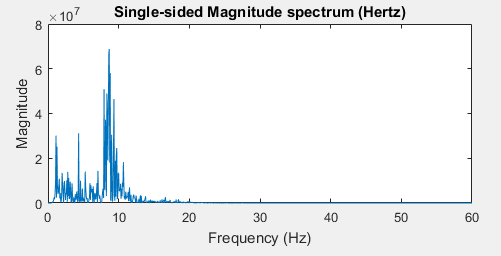
- with/without digital filter?



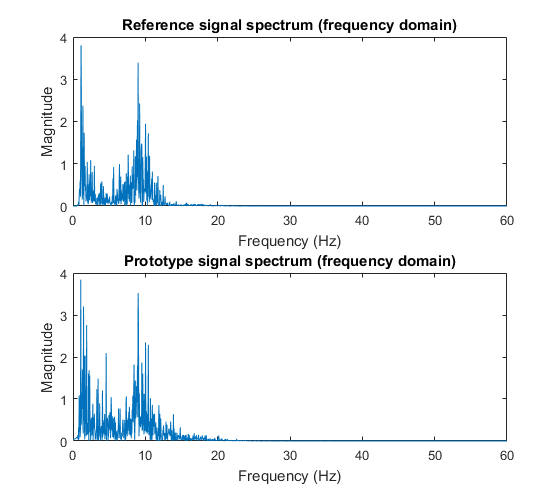
**Figure 7**: *Raw sample of eye opening and closing evoked potentials (EPs).*

**Figure 8**: 8A.: *Frequency spectrum of readings of EEG signal of a subject with opened eyes. 8B.: Same spectrum with closed eyes.*

**8A.**



**8B.**

**

*Comparison tests*

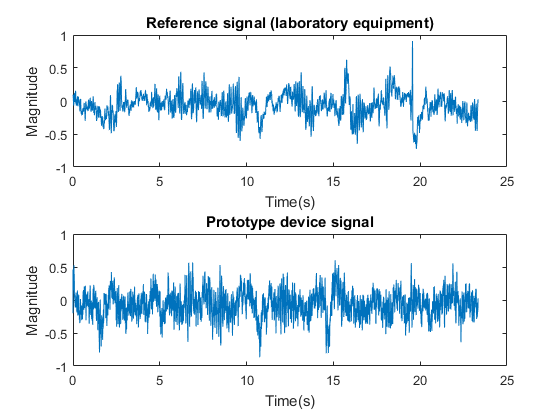
Figure 9 shows the comparison between the signals acquired by our prototype device and a high-end device used in the laboratory for research. The comparison was made using only one channel. Both signal electrodes were put near the Cz zone, with 1 centimeter of distance from each other on the scalp. The negative electrodes were put near them, while the ground electrodes were put on each mastoids.

The acquisition started unsynchronized, and the data was synchronized later by visual inspection, using the evoked potential caused by the opening of the eyes as reference, which was clear on both data sets.

By using correlation coefficient in MATLAB, we can compare the samples. This coefficient can vary from -1 to 1, and indicates the relation between two signals. 1 being the maximum positive correlation, 0 meaning no correlation, and -1 the maximum negative correlation (inverse proportion).

In the raw samples pictured in Figure 9, the coefficient extracted was only about 0.272. But comparing the signals after using FFT transform brings the coefficient as high as 0.688 (See Figure 10). We can conclude that, although the raw signal acquired was relatively different due to interference and distance in scalp, the correlation was strong in the frequency domain, which is more relevant for feature extraction in general.

(develop and explain about a simple application, for example, lighting a LED by blinking/closing eyes)



**Figure 9**: *Comparison of prototype and reference raw signals with signal electrodes around 1cm apart from each other. The correlation coefficient extracted was 0.0272.*

**Figure 10**: *Comparison of same measurements from Figure 9, but in frequency domain. The correlation coefficient extracted was 0.688, a significant improvement.*

**Conclusions and prospects**

We can conclude, by seeing the results, that although there are many limitations regarding the quality of the samples acquired, it is possible to extract features and information from our device.

It should also be considered that the differences in the signal comparison might occur due to the physical distance between the electrodes in the scalp.

**8B.**

Further addition of channels can be done without many changes in the hardware, and with some changes in the software.

EEG devices still are very delicate to handle, due to its high interference sensibility and the complexity of diminishing impedance (mainly avoiding the hair and cleaning the scalp). So there is still much work to do to allow home appliances to support this technology.

**Bibliography**

(gather and insert references)